

Viviane Wolters

**Promoting Cooperative Behavior
with Collective Decision-Making
in Highway-Based
Multi-Agent Pathfinding**



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF

FAKULTÄT FÜR
INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

**Promoting Cooperative Behavior
with Collective Decision-Making
in Highway-Based
Multi-Agent Pathfinding**

Master Thesis

Viviane Wolters

November 6, 2021

Supervisor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Advisor: Sebastian Mai

Viviane Wolters: *Promoting Cooperative Behavior
with Collective Decision-Making
in Highway-Based
Multi-Agent Pathfinding*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2021.

Abstract

Multi-Agent Pathfinding concerns finding paths for multiple agents, from their starting points to their target destinations, in a manner that enables the agents to move simultaneously without colliding with obstacles or other agents.

This thesis presents a new approach for multi-agent pathfinding on grid maps. This approach utilizes an algorithm based on A*-search and prevents collisions through a binding highway layout created by several collective decisions of the agents. Specifically, the problem topology is considered as a graph with nodes for each grid cell and undirected edges between adjacent cells. In each decision round the agents vote for an edge to be added to the highway layout. They determine their opinion about it using an A*-search, which is influenced to return paths that contain edges of the emerging highway graph to some degree. Additional rules ensure that the resulting layout will be designed in a way that no collisions are possible when moving along the highways. At the end the decision rounds result in a highway layout that is largely suitable for all agents. To guarantee that each agent arrives at its destination, agents are allowed to change the highway layout during path execution if the other agents agree.

An analysis of the A*-based search on collectively created highways has shown that the choice of parameters to affect the A*-search is a tradeoff between the quality of the highway layout, the sum of cost or makespan, and the speed of the algorithm. However, with a good parameter choice, the algorithm scales significantly better than a constraint-based search. For large maps as well as maps with a lot of free space, it also scales better than a simple priority-based planner. Nevertheless, this approach shows minor weaknesses for warehouse-like domains.

Contents

List of Figures	V
List of Tables	VII
1 Introduction	1
2 Basic Principles and State of the Art	3
2.1 Multi-Agent Pathfinding	3
2.1.1 Formal Description	3
2.1.2 Objective Functions	5
2.1.3 Overview of Multi-Agent Pathfinding Solvers	5
2.1.4 Highway-Using Approaches	7
2.2 Collective Decision-Making	8
2.2.1 Aggregation Methods	8
2.2.2 Collective Decisions in Multi-Agent Systems	9
3 Methodology	13
3.1 Definitions	13
3.2 The Algorithm: A*+CCHWY	16
3.2.1 Path Planning	16
3.2.2 Path Execution	20
4 Experiments	25
4.1 Implementation	25
4.2 Problem-Based Benchmark Suite	27
4.3 Experimental Design	29
4.3.1 Experiment 1: Parameter analysis	29
4.3.2 Experiment 2: The Algorithm Performance	30

4.4	Discussion of Results	31
4.4.1	Experiment 1: Parameter Analysis	31
4.4.2	Experiment 2: The Algorithm Performance	39
5	Conclusion and Outlook	43
	Bibliography	45

List of Figures

2.1	Four Types of MAPF Conflicts	4
3.1	Graph Structures	14
3.2	Movement Restrictions	15
3.3	Quality difference of varying highway layouts	17
3.4	Opinion Definition	18
3.5	Side Effects of Adding Edges to the Highway Graph	21
3.6	Example of Deadlock Situation	22
4.1	UML-Diagram of the Simulation Model	26
4.2	Results of Experiment 1.2 and 1.4: No. of Highway Changes	32
4.3	Results of Experiment 1.2 and 1.4: Sum of Cost, Makespan and No. of Deadlocks	34
4.4	Results of Experiment 1.1: Success-Rate	36
4.5	Results of Experiment 1.2: Success-Rate	37
4.6	Results of Experiment 1.2 and 1.3: No. of Decision-Rounds	37
4.7	Results of Experiment 2.1 and 2.2: Success Rate, Sum of Cost, Makespan	40

List of Tables

2.1	Overview of Popular MAPF Solvers	6
4.1	Characteristics of maps used in the benchmark	28
4.2	Experimental design of Experiment 1: Parameter Analysis	42
4.3	Experimental design of Experiment 2: Algorithm Quality	42

1 Introduction

Multi-agent pathfinding (MAPF) is a research topic assigned to the broad field of artificial intelligence. It deals with planning paths for a group of agents that have to move simultaneously from their starting to their target positions without colliding with existing obstacles or the other agents.

MAPF is relevant to any real-world application involving several moving agents, such as warehouse management, airport towing, autonomous vehicles, robotics, and video games. Due to its relevance in highly diverse areas, many scientific works have been published, dealing with various aspects and approaches to solving such planning problems.

Many of these approaches concentrate on optimally solving MAPF problems by minimizing objective functions such as the sum of cost. However, solving such problems optimally has been shown to be NP-hard [49] and is therefore only suitable for a small number of agents. To make pathfinding feasible for a wider group of agents, some approaches have thus renounced optimal solutions. For example, Cohen et al. [9][10] use a highway layout and Jansen et al. [19] direction maps as influencing schemes to manipulate the heuristic search while minimizing conflicts. Jansen et al. [18], in turn, use flow restrictions to reduce the search space.

In one way or the other, all three approaches influence the heuristic search to return paths orientating on a scheme of moving directions or following flow restrictions. This, in turn, leads to an implicit global behavior of the agents, which reduces the conflicts to be solved and prevents collisions. Nonetheless, to create an effective influencing scheme or suitable flow restrictions for a given environment, these approaches are dependent on humans expertise or a learning algorithm running beforehand.

In this thesis a new concept will be analyzed, where such an influencing scheme is a result of several collective decisions made by the agents during the path-planning process and on the fly while navigating towards their destinations.

This thesis aims to develop an algorithm capable of solving a MAPF problem on a grid map so that the involved agents decide collectively about the possible direction of motion for each grid cell for the purpose of collision avoidance. In addition to its design and implementation, the algorithm has to be analyzed and compared to other algorithms within different benchmark scenarios to answer the following questions:

1. How does the algorithm's parameter influence its performance?
2. How does the algorithm perform in comparison to other algorithms?
3. Where are the weaknesses and limits of the algorithm?

The remainder of this thesis is structured as follows: Chapter 2 presents state of the art and is divided into two parts, MAPF and collective decision-making (CDM). Each section provides formal definitions and an overview of existing approaches. Chapter 3 presents the used methodology and Chapter 4 the conducted experiments, including a description of the simulation used for it, an explanation of its design, and an analysis of the results. The last chapter answers the four previously mentioned research questions and contains recommendations for further research.

2 Basic Principles and State of the Art

The objective of the algorithm presented in this thesis is to solve a MAPF problem with the help of agents' collectively designed movement restrictions called highways. This chapter explains the theoretical foundation of the two key aspects MAPF and CDM. Furthermore, it provides an overview of their state of the art and highlights the commonalities and differences of existing approaches in comparison to the one presented here.

2.1 Multi-Agent Pathfinding

MAPF entails finding and planning paths between the specified start and target positions of a group of agents. Specifically, all agents need to be able to execute their path simultaneously and reach their target without colliding with each other or with static obstacles.

2.1.1 Formal Description

Stern et al. [38] formally describe a classical MAPF problem with k agents as a tuple $\langle G, s, t \rangle$, where

$$G = (V, E) \tag{2.1}$$

is an undirected graph, which vertices V are possible locations of the agents, and which edges $(n, n') \in E$ represent the agents possibility of moving from vertex n to n' without passing through any other vertex.

The second parameter of the tuple,

$$s: [1, \dots, k] \rightarrow V, \tag{2.2}$$

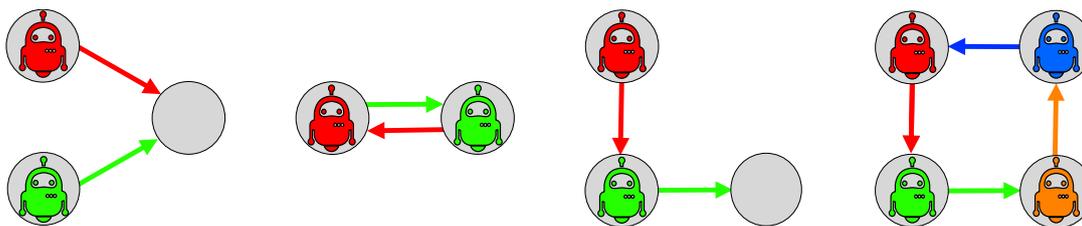


Figure 2.1: Four types of MAPF conflicts. From left to right: vertex conflict, edge conflict, following conflict, cycle conflict

is a function that maps an agent to its initial location.

The last parameter, on the other hand, that is,

$$t : [1, \dots, k] \rightarrow V, \quad (2.3)$$

is a function that maps an agent to the location of its target destination.

The time in all classical MAPF problems is assumed to be discrete. At each time step, an agent can perform one single action, which can be either moving from the current to one of the adjacent locations or waiting at the current location. Stern et al. [38] denotes a sequence of this actions $\pi = (a_1, \dots, a_n)$, leading one agent i from its initial position $s(i)$ to its target position $t(i)$, as "single-agent plan". Formally:

$$t(i) = a_n(\dots a_2(a_1(s(i)))\dots) \quad (2.4)$$

Combining these "single-agent plans" from each of the k agents to a joint plan without conflicts leads to a valid solution for the MAPF problem. Figure 2.1 displays the following four possible conflicts:

- **Vertex conflict:** Both agents plan to occupy the same vertex at the same time.
- **Edge conflict:** Two agents plan to swap locations over the same edge at the same time.
- **Following conflict:** An agent plans to occupy a location at time step $t + 1$ that was occupied by some other agent at time step t .
- **Cycle conflict:** Like the following conflict but with more agents simultaneously in a circular pattern.

In most MAPF scenarios, it is enough to solve the first two of the mentioned conflicts to obtain a valid solution; however, some applications have stricter requirements and consider other conflicts, such as the two last mentioned conflicts.

2.1.2 Objective Functions

A MAPF problem might have more than one valid solution. Therefore, most applications aim to solve a given MAPF problem by finding a solution that optimizes an objective function. The most common ones are makespan and sum of costs [38].

The makespan M of a joint plan $\Pi = \{\pi_1, \dots, \pi_k\}$, is the number of time steps until all agents reach their destinations [36][40], formally described as:

$$M(\Pi) = \max_{1 \leq i \leq k} |\pi_i| \quad (2.5)$$

On the other hand, the sum of costs SOC , also known as flowtime, is the sum of time steps each agent requires to reach its destination [36][40], formally described as:

$$SOC(\Pi) = \sum_{1 \leq i \leq k} |\pi_i| \quad (2.6)$$

2.1.3 Overview of Multi-Agent Pathfinding Solvers

MAPF is an extensively studied field of research in which numerous different solving techniques were developed in the past years. These solvers can be roughly categorized into optimal and suboptimal ones.

Optimal solvers guarantee to return a solution that minimizes an objective function (see Chapter 2.1.2). Stern et al. [37] classify four types of optimal solvers: the first class consists of A*-based algorithms that search the k-agent search space using an adaption of the A*-algorithm.

Then, there is the increasing cost tree search, which splits MAPF into two problems: first, finding the cost added by each agent, and second, finding a valid solution with these costs.

Another group are conflict-based searches (CBS), which solve a sequence of single-agent pathfinding problems by incrementally adding specific constraints to them in a way that ensures completeness and optimality.

Optimality	Category	Algorithm	Literature
Optimal	A* Extensions	ID	[36]
		OD	[36]
		EPEA*	[16]
		M*	[44]
	Increasing Cost Tree	ICT	[34]
	Conflict Based	CBS	[33]
		ICBS	[5]
	Reduction Based	(MDD-)SAT-solver	[41]
		CSP-solver	[30]
		ASP	[13]
ILP		[48]	
Bounded-Suboptimal	Search Based	CBS+HWY	[10]
		ECBS(+HWY)	[3] [10]
		EECBS	[23]
		HCA*	[35]
		WHCA*	[35]
Suboptimal	Rule Based	Push-and-Swap	[25]
		Push-and-Rotate	[12]
		TASS	[21]
		BIBOX	[4]
	Hybrid	A* + DMs	[19]
		FAR	[18]
		MAPP	[45]

Table 2.1: Overview of popular MAPF solvers

Finally, there are reduction-based approaches, which reduce MAPF to a propositional satisfiability problem (SAT), a communicating sequential process (CSP), or similar. These, in turn, can be solved with already existing high-quality designed algorithms.

Suboptimal solvers aim to rapidly find paths for all agents and are therefore often a suitable alternative for MAPF problems that have to be scalable for a high number of agents. Felner et al.[14] classify them into search-based, rule-based, and hybrid algorithms:

Suboptimal search-based algorithms are often based on modifications or relaxations of optimal MAPF algorithms. Some of them are bounded-suboptimal, which means that they offer shorter solving times by sacrificing optimality within a desired factor of optimal.

Algorithms that include specific agent-movement rules for different scenarios rather than massive search belong to the group of suboptimal rule-based approaches.

Hybrid approaches, in turn, comprise both movement rules and search.

Table 2.1 offers an overview of some popular MAPF approaches. They are classified into the described categories, with information about optimality and completeness.

2.1.4 Highway-Using Approaches

Optimally solving MAPF problems is known to be NP-hard [49]. Optimal planners are therefore only feasible for a low number of agents. One solution for this issue is offered by the previously mentioned bounded-suboptimal algorithms. The enhanced conflict base search (ECBS) is the fastest of these, but it becomes slower as the number of conflicts increases in a given scenario. Hence, Cohen et al. [10][9] invented a variant of CBS and ECBS that uses a highway layout to reduce the number of conflicts, thus speeding up the algorithms. The highway layout works like a simple influencing scheme based on the ideas behind experience graphs [28] to derive new heuristic values, which encourage the search algorithm to return paths that include the edges of the defined highways. This, in turn, encourages a global behavior of the agents moving alongside the highway layout, which avoids collisions.

Cohen et al. [9][10] were not the only researchers who used highways for their MAPF approach. Jansen and Sturtevant [19] introduced a method that uses di-

rection maps to influence heuristic search. These direction maps are created by an algorithm learning the movements of the agents in a specific world. The finished maps can then be used to modify the planning process of any heuristic search by changing the underlying edge costs of the search graph, encouraging the search algorithm to return paths that correspond to the direction map. This, in turn, leads to an implicit cooperative behavior of the agents similar to the approach of Cohen et al. [10][9].

Wang and Botea [18] also use the idea of highways. The first step of their approach is to impose flow annotations on the given map and create a directed search graph instead of the usually used undirected one. The flow annotation restricts the traffic on each row and column to only one direction. Alternating rows have the same horizontal direction, and alternating columns have the same vertical direction. A highway layout emerges by covering the given map with criss-crossing virtual roads. Additional rules are provided to ensure that no feasible path becomes infeasible.

Like the first two mentioned approaches, the method presented in this thesis also uses an influencing schema of movement directions to manipulate heuristic search to return paths that lay within the scheme's framework referred to as highway layout. The difference lies in the process of creating the highway layout. The highway layout is neither designed by humans nor by a learning algorithm; rather, it results from several collective decisions made by the agents.

2.2 Collective Decision-Making

Making a decision means selecting among alternatives. A collective decision-making system uses an aggregation mechanism to combine the input of several individuals to generate a common decision.

2.2.1 Aggregation Methods

There are many ways to aggregate individual preferences to a collective preference or outcome.

Let N be a set of individuals and A a set of alternatives; denote the set of linear orders on A by $L(A)$, where the preferences are assumed to be elements of $L(A)$.

Taking these assumptions into account, a formal description of an aggregation method, also called social-choice rule, can be described as:

$$K : L(A)^N \rightarrow A \tag{2.7}$$

Thus, a social-choice rule K assigns to each profile $p \in L(A)^N$ a collective choice $K(p)$ in A . Harrie de Swart et. al. [11] pointed out some of the more frequently occurring choice rules, which are:

- Plurality rule: The alternative which polls more than any other is elected.
- Majority rule: If there is an alternative x that defeats every other one in pairwise comparison, this alternative x must win.
- Borda rule: Points are assigned to the alternatives based on their ranking: one point for the last choice, two points for the second-to-last choice, and so on. The point values are summed, and the one with the highest total points is the winner.
- Approval Voting: The voter can divide the alternatives into two classes: the ones that the agent approves and the ones that the agent disapproves. The number of alternatives that are found to be acceptable depend on the agent. The one that gets the most votes is the winner.

2.2.2 Collective Decisions in Multi-Agent Systems

CDM in artificial systems has been inspired by behavior studies on self-organizing animal groups. These studies range from unicellular organisms [46] [29], to social insects [32] [7], fish schools [24], and groups of mammals [22][39]. The behavior of humans has also been studied [1] [2].

Brambilla et al. [6] distinguished CDM systems into two categories: consensus achievement and task allocation. Consensus achievement concerns a behavior whereby several agents make a decision on a given matter, whereas in task allocation the agents allocate themselves to different tasks to maximize collective performance and reach a common objective. Valentine et al. [43] further divide consensus achievement into discrete and continuous problems. This classification depends on the cardinality of the choices an agent can make. Specifically, choices can either be finite and countable — in which case one can speak of a discrete problem — or infinite and measurable, which would be assigned to a

continuous problem.

The algorithm presented in this thesis can most likely be assigned to a discrete consensus achievement problem, which is the reason why the following focuses on such approaches only.

Wessnitzer and Melhuish [47] solve with their approach a target hunting scenario with moving preys. In the beginning, each agent favors a prey chosen at random. At each time step, the agents apply the majority rule over their neighborhood to reconsider and possibly change their opinions, which leads to a swarm that decides which prey to hunt first.

Parker and Zhang [27] examine a scenario where the agents have to discriminate between two sites with different brightness levels and begin in either a searching state or an idle state. Those in the searching state search for alternatives, whereas the idle agents wait to be recruited into the process. When an agent finds an alternative, it determine its quality, transit to an advocating state, and rejoins its teammates. Advocating agents frequently send recruit messages to their teammates, and the frequency increases with the options' quality. Agents in the idle and advocating states are recruitable. When such an agent receives a recruit message, it reenters the researching state, where it evaluates the quality of the specified alternative and then enters the advocating state favoring it. Meanwhile, agents estimate the popularity of their favored option and use this information to test if a quorum has been reached. Agents in the committed state send commit messages to those that they encounter. When a robot receives such a message, it commits to the specified alternative, which happens until a complete consensus is achieved.

Schneider et al. [31] consider the well-known double bridge scenario from Goss et al. [17] where a swarm has to find the shortest of two paths between two locations. Initially, the robots of Schneider et al.'s experiment have their own opinion about the quality of the paths. While traveling between the two locations, the agents encounter other agents, observe their opinions, and store them into a memory of size k . In the case of full memory, the oldest opinion is replaced by the new one. The agents change their current opinion after successively meeting k agents only favoring these specific other option. This behavior eventually leads to consensus on one single opinion. Due to a bias induced by the different travel time, high probability consensus most likely falls on the opinion representing the shortest path.

The mentioned approaches solve different problem scenarios, but they all share a similar rational structure and therefore can be grouped into the same framework, which is called best-of-n problem. Valentine et al. [43] defines the best-of-n-problem as follows: given a swarm of N agents, a swarm has found a solution to a particular instance of the best-of-n problem as soon as it makes a collective decision for any option $i \in 1, \dots, n$. A collective decision is represented by the establishment of a large majority $M \geq (1 - \delta)N$ of agents that share the same preference for a certain option i , where δ , $0 \leq \delta \ll 0.5$, represents a tolerance threshold set by the designer. In the case $\delta = 0$, the swarm has reached a consensus decision where all agents favor the same option i .

Strategies for solving best-of-n problems consist of three components: an exploring and a dissemination mechanism and a decision rule [42].

The exploring mechanism is needed to discover possible alternatives of the best-of-n problem and gather sample estimates of the associated quality. After sampling the quality of an option, the agent transits to a dissemination state where it shares the opinion within the rest of the swarm. Disseminating an opinion can be as simple as broadcasting it locally towards other members of the swarm. The duration of the dissemination influences the collective decision and is therefore proportional to the quality of the advertised option. Options with higher quality are advertised for a longer time and have higher chances of being heard by other members of the swarm. During the dissemination, the agents also listen to the opinions of others and keep track of their frequency. Before terminating the dissemination period, the agents apply a decision rule to reconsider and possibly change their own opinion. Popular decision rules are represented by the voter model, with which an agent adopts the opinion of a randomly chosen neighbor, and the majority rule, with which the agent adopts the opinion shared by the majority of its neighbors. Other decision rules aggregating different preferences to a single one, such as the methods mentioned in Chapter 2.2.1, are also possible.

The approach presented in this thesis cannot be classified as a traditional best-of-n approach like the ones described above. None of the mentioned approaches can be compared with the one of this thesis. One reason is that the opinion of each agent about the best option is neither randomly chosen nor a result of an environment exploration since the environment is already known and the opinion is based on a heuristic search. Another difference is that the agents share their opinions with all other agents in the scenario rather than only their neighborhood. Furthermore, all of the above-mentioned approaches are based on a swarm of agents which share similar preferences. In general, the agents in this

thesis share the same goal of finding a way from an initial position to a destination, but both the initial position and the destination are assigned individually to each agent, which could lead to situations where preferences regarding the design of the highways on the map massively differ because the quality of each decision option may be different for each agent. It should also be mentioned that most of the recent best-of-n approaches have focused on decision problems with just two options to decide from. There are just a few studies (i.e., [15] and [31]) that consider more options, such as the approach used in this thesis.

3 Methodology

3.1 Definitions

The problem solved by the algorithm introduced in this work can be defined as a classical MAPF problem [37] described formally in 2.1.1. Further elements about solving the MAPF problem with a collectively created highway layout are specified in the following.

Environment and search graph:

The problem topology is represented as a two-dimensional grid map, which means that the map of the agents' environment is evenly discretized into a grid of cells similar to a chess field. Specifically, one cell has eight neighboring cells, four cardinal and four diagonal. Each cell is either traversable or blocked by a static obstacle. Hence, a search graph is created with one vertex for each traversable cell and an undirected edge between two traversable neighboring cells. One exception is made: there is no edge between two diagonal cells separated by one or two blocked cells since it would not be physically possible for an agent to move from one to the other cell without colliding with these obstacles (compare with Figure 3.1)

Highways:

To avoid collisions and solve conflicts, the agents have to decide collectively about binding movement restrictions in the form of so-called highways. Each traversable cell can be a part of one or more highways. Each cell within a highway has specified moving directions. Relating to the eight possible neighbor cells, also eight directions are theoretically conceivable.

A highway layout can host one contiguous or multiple separated highways. In the viewpoint of the search graph G all highways combined as layout can be described as a subgraph G_H defined as:

$$G_H = (V_H, E_H) \quad \text{where} \quad G_H \subset G \quad (3.1)$$

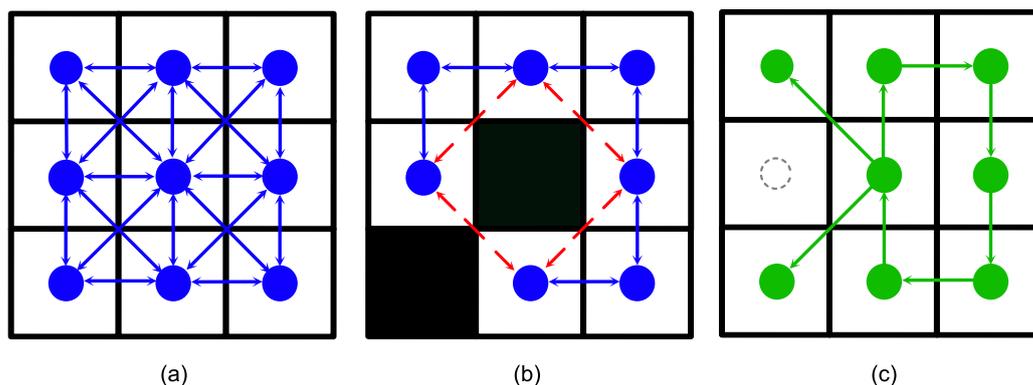


Figure 3.1: (a) Search graph of an empty grid map. (b) Search graph of a map with obstacles. (c) Possible design of a highway graph as sub graph of (a)

Figure 3.1c shows an example of a valid highway graph. Valid means that this graph satisfies the following two rules to prevent edge and vertex conflicts during movement:

1. Bidirected edges are not allowed
2. A vertex can have multiple successors but only one predecessor

Legal moves:

Like in the formal MAPF description, at every discrete time step, each agent can either wait in the currently occupied cell or move to a neighboring one. Furthermore, moving is only possible if a corresponding highway edge E_H exists and no waiting agent is occupying the next cell. Moreover, diagonal movements are possible even if both cells that separate the starting cell from the target one are occupied by waiting agents. It is assumed that the agents are small enough not to collide in this situation. Furthermore, two agents can move diagonally if their paths would crisscross each other since this could be easily solved with a right-for-left rule or similar in real life. All described scenarios can be comprehended in Figure 3.2

Further assumptions:

After an agent reaches its target, it disappears from the map. When all agents have disappeared, the problem is solved.

Furthermore, homogenous agents and uniform speed, as well as complete and faultless communication between the agents for all decision-making processes, are assumed.

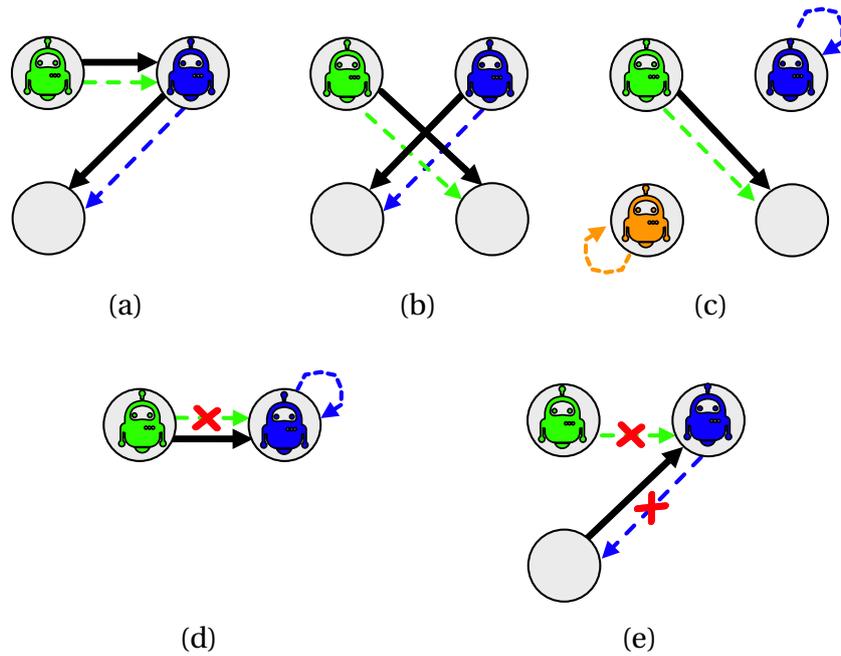


Figure 3.2: Highway edges are represented as black arrows. Intended movements are marked as dotted colored lines. (a) Legal move: Following is allowed. (b) Legal move: Crisscrossing paths are allowed. (c) Legal move: Agents are assumed to be small enough not to collide. (d) Illegal move: Vertex conflict because of a waiting agent. (e) Illegal move: No or wrong highway edge.

3.2 The Algorithm: A*+CCHWY

The following described algorithm, named A*+CCHWY, runs separately on each agent and can be divided into two different parts:

1. Path planning
2. Path execution

The path planning process takes part before the agents start to move. The objective is to find a path for each agent from a starting to a target position and create alongside an initial highway layout by making several collective decisions. The path planning process ends and the plan execution begins when there are no decisions left.

During plan execution, the agents start to move on the highways towards the goal. Each agent is allowed to change the highway layout on the fly by requesting permission from the others. The plan execution — and, therefore, the algorithm — terminates if an agent reaches its goal. The following chapters describe the path planning and plan execution further.

3.2.1 Path Planning

Algorithm 1 shows the single steps of the path planning process through which each agent proceeds. It consists of several collective decision rounds on particular edges of the search graph G to be added to the highway graph G_H .

The final result of these rounds of decision-making should be an effective highway layout. The example in Figure 3.3 illustrates the difference in quality between two varying highway layouts for the same problem scenario. Accordingly, in this work, a proper highway layout is defined as one which has the property of not needing to be changed during plan execution or needs to be changed as little as necessary. Additionally, a decent highway layout ensures that agents take only as large detours as necessary and as small detours as possible.

Opinion Definition:

Creating an effective highway design begins with each agent forming an initial opinion about which edges should be included in the highway graph. These are the edges of the path an A*-search returns.

To encourage cooperative movement, the opinion of an agent is able to change iteration by iteration within the path planning process due to a growing highway graph. Every edge added to the highway graph that is not compatible with

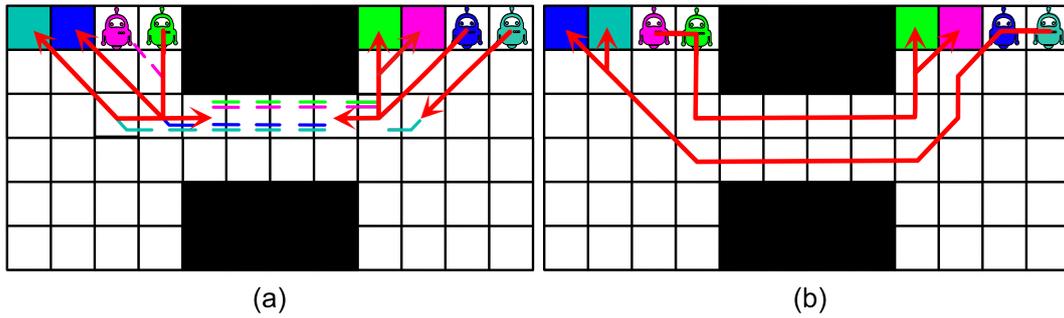


Figure 3.3: Highways are represented as red arrows. Paths of the agents not included in the highway design are marked as dashed lines in the color of the regarding agent. (a) Bad quality: None of the agents can move to their goals without changing the existing highway layout. (b) Good quality: All agents can move to their goals without changing the highway layout.

Algorithm 1 Path Planning Process

```

1: procedure PATH_PLANNING()
2:   my_path ← ASTAR(graph, hwy_graph, w_penalize, w_reward, start, goal)
3:   repeat
4:     my_opinion ← DEFINE_OPINION(my_path, highway_graph)
5:     SEND_OPINION_TO_OTHERS(my_opinion)
6:     open_decisions ← RECEIVE_OTHERS_OPINION()
7:     highway_edge ← MAKE_DECISION(open_decisions)
8:     UPDATE_GRAPH(highway_graph, highway_edge)
9:     if my_path contradicts highway_edge and last search successful then
10:      path ← ASTAR(graph, hwy_graph, w_penalize, w_reward, start, goal)
11:    end if
12:    if path is not empty then
13:      my_path ← path
14:    end if
15:  until open_decisions is empty
16:  return my_path
17: end procedure

```

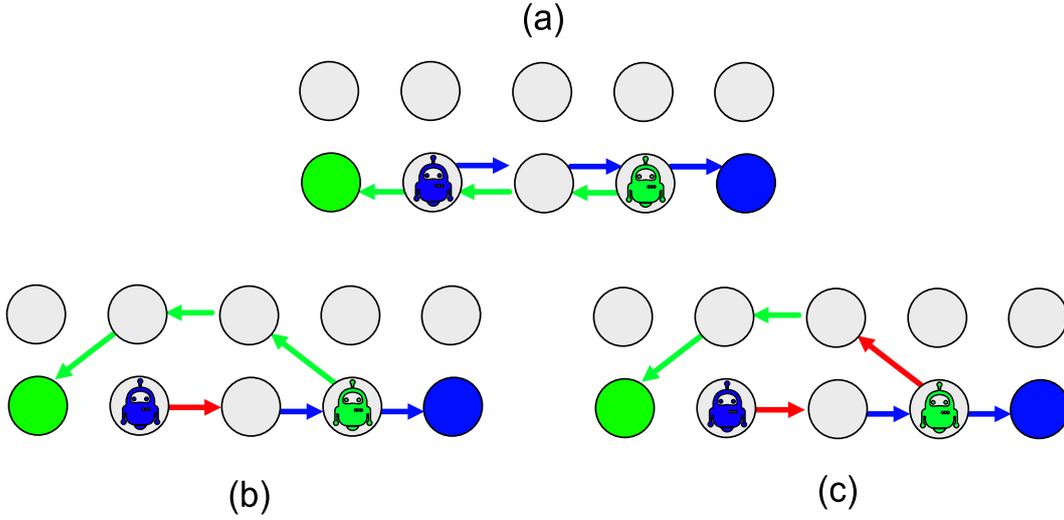


Figure 3.4: Highways are represented as red arrows. Paths of the agents not included in the highway design are marked as colored lines (a) Initial opinion of the agents. (b) Green agent has to replan. Added highway edge does not fit to its current opinion. (c) No replanning needed.

the previously planned path leads a agent to replan its paths and, therefore, to change its opinion 3.4). If the replanning is not successful, the agent will stay with its old path and opinion. Replanning — and, therefore, opinion changing — is enabled by influencing the A*-search with the previously taken collective decisions. The higher the influence, the larger should be the number of edges of the highway graph within the returned path.

Described in more detail: a highway graph as an addition to the standard search graph can be used to modify the planning process of any heuristic search by changing the underlying costs of traversing the world instead of modifying the search algorithm itself. For instance, A* expands nodes in order of the f-cost, where,

$$f(n) = g(n) + h(n). \tag{3.2}$$

Weighted-A* adds a weight to the heuristic function $h(n)$ in order to encourage expanding nodes with lower heuristic cost. In this approach, the opposite is done. Inspired by the idea of Jansen and Sturtevant [19] the cost of traversing an edge $g(n)$ is modified. Movements that are congruent with edges in the highway graph are encouraged, whereas movements in the opposite direction

are discouraged. Furthermore, movements leading to a cell whose vertex already has a predecessor within the highway graph are discouraged. The exact reward is a parameter w_r and the exact penalty is a parameter w_p . With an underlying edge cost of c , this leads to the following equation for the g-cost:

$$g(n) = \sum_{(n_i, n_{i+1}) \in E} \begin{cases} c \cdot w_r & \text{for } (n_i, n_{i+1}) \in E_H \\ c \cdot w_p & \text{for } (n_{i+1}, n_i) \in E_H \\ c \cdot w_p & \text{for } n_{i+1} \in V_H \wedge (n_x, n_{i+1}) \in E_H \\ c & \text{otherwise} \end{cases} \quad (3.3)$$

The octile distance function was chosen for the heuristics. The octile distance of two vertices $n = (x, y)$ and $n' = (x', y')$ is as follows:

$$h(n) = d(n, n') = \max(\Delta x, \Delta y) + (\sqrt{2} - 1) * \min(\Delta x, \Delta y) \quad (3.4)$$

where $\Delta x = |x - x'|$ and $\Delta y = |y - y'|$

Opinion dissemination:

After forming an opinion, the agents send it to all other agents. The content of the messages can be described as a ballot within which the agents vote for edges that should be added to the highway graph. In every decision round, each agent has one vote for each vertex of the search graph that is not already part of the highway graph or, if it is, does not have a predecessor. With this vote, an agent can namely elect one out of eight possible predecessors for the corresponding vertex.

Each of the agents' votes is weighted. The score of the vote depends on the heuristic distance between the agents' start vertex and their vertex of interest. In consequence, votes for vertices nearer to the agents' start vertex will obtain a higher weight than votes for vertices with a greater distance.

The calculation of the weight w_d of a vote for a specific vertex n' with the agents start vertex n , the octile distance function $d(n, n')$ (see Equation 3.5) and the maximum possible node distance D , can be formally described as:

$$w_d = w(n, n') = D - d(n, n') \quad (3.5)$$

Agents abstain from voting for vertices not included in their previously planned path and about which, therefore, they do not have an opinion. Furthermore,

votes for edges that do not satisfy the two highway graph rules defined in Chapter 3.1 when adding to the highway graph are not valid; hence, they are not included within the ballot.

Decision-making:

After receiving the ballots of all other agents, the decision-making takes place. One minority rule and a consecutive plurality rule aggregate the votes to a joint decision about the edge (n, n') to be added to the highway graph. First, the minority rule determines n' by selecting the cell with the least abstained votes. If a tie occurs, the vertex with the least divergent votes for a specific predecessor is chosen. If this does not break the tie, the winner is chosen randomly. Finally, the plurality rule identifies the predecessor n for the previously selected vertex. Once again, if a tie occurs, the winner is selected randomly. The winner edge is then added to the highway graph.

3.2.2 Path Execution

Algorithm 2 shows the process for navigating the agents from their starting to their goal position. Its single steps are explained in detail in the following.

Requesting changes:

During every iteration, each agent first checks if it needs to change the highway design in order to move to the next cell of its path. Considering r as a Boolean variable that defines whether an agent must request a change within the highway graph ($r = True$) or not ($r = False$), the following equation is obtained:

$$r(n) = \begin{cases} True & \text{for } (n_i, n_{i+1}) \notin E_H \\ False & \text{otherwise} \end{cases} \quad (3.6)$$

No matter if an agent is in need of a highway modification or not, it will send a message to all the others within the next step. This message includes information about the agent's position, its next target and, if needed, the requested edge to be added to the highway graph and the number of steps an agent could take after the requested change until a new one is needed. This number is infinite when no more changes are needed to reach the goal.

Deciding on the others' requests:

After an agent receives all the messages, it starts deciding on open requests. An agent can either agree or disagree with a request depending on what side effects

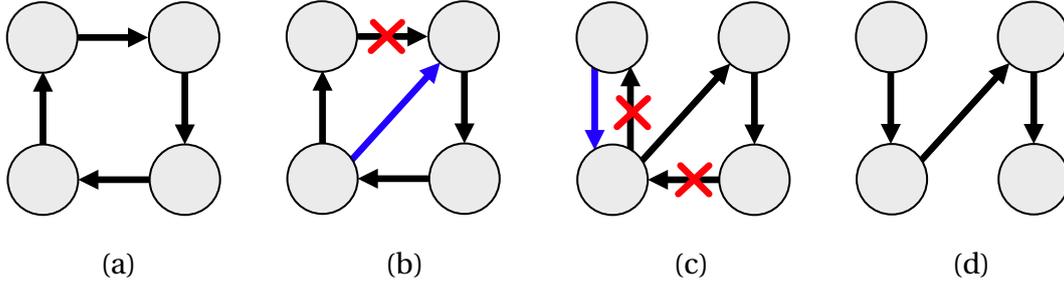


Figure 3.5: Possible side effects of adding an edge to the highway graph. Edges within the highway graph are marked as black arrows. Edges to be added to the graph are marked as blue arrows. (a) Initial highway layout. (b) Edge is removed because of vertex conflict. (c) Edges are removed because of an edge and a vertex conflict. (d) Highway graph after adding the two edges

the desired edge would cause when added to the highway graph. The ability to disagree with a request is important due to the fact that adding an edge to the original highway graph also means to remove existing edges that would conflict with the new one according to the rules indicated in Chapter 3.1. Thus, an agent disagrees with a request if it will affect a highway edge that it needs in order to move to its next target. Figure 3.5 shows an example of possible side effects.

Let the Boolean variable a be the answer of an agent deciding whether to agree ($a = True$) or disagree ($a = False$) to a request. Let (n_r, n'_r) be the requested edge and n_i be the current position of the deciding agent. Accordingly, the result of a is defined as follows:

$$a(n_r) = \begin{cases} False & \text{for } (n_r, n'_r) = (n'_i, n_i) \wedge (n_i, n'_i) \in E_H & \text{Edge Conflict (1)} \\ False & \text{for } (n_r, n'_r) = (n'_i, n_i) \wedge x_r \leq x_i & \text{Edge Conflict (2)} \\ False & \text{for } n'_r = n_i \wedge (n_i, n'_i) \in E_H & \text{Vertex Conflict (3)} \\ False & \text{for } n'_r = n_i \wedge x_r \leq x_i & \text{Vertex Conflict (4)} \\ True & \text{otherwise} & \text{No Conflict (5)} \end{cases} \quad (3.7)$$

Updating the highway graph:

After deciding on all requests, the agent sends the results to the others and receives their opinions in return.

To grant a request, all agents have to agree to it. If only one agent decides against

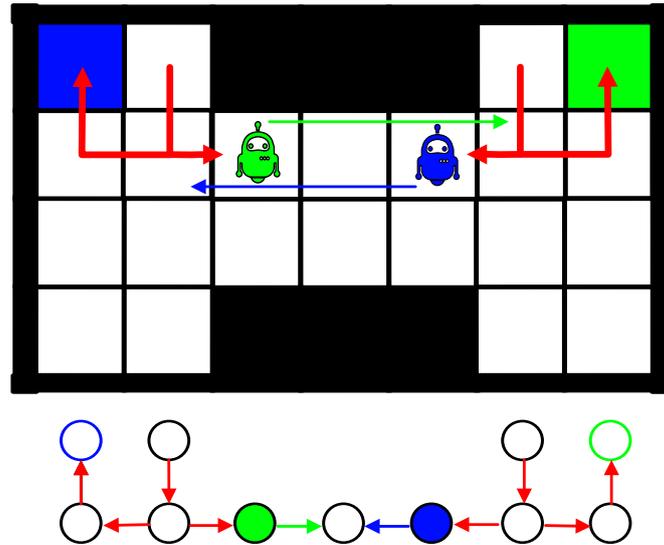


Figure 3.6: Map and highway graph of a deadlock situation. Each agent is waiting for the other's approval to add the requested edge to the highway graph. According to Equation 3.7.4 both will disagree on the request.

the change, it will not take place. Subsequently, the highway graph will be updated with the requests which have reached a consensus agreement.

After updating the highway graph, the agents can finally move one step ahead on their path if it is possible.

Deadlocks:

During path execution, the agents sometimes have to wait at their current position because a desired change in the highway layout has not been approved by the others. Such situations can lead to deadlocks, which can be seen as circular wait situations as defined in Coffman et al. [8]. Figure 3.6 shows an example of such a deadlock. The agents are waiting for the other's approval, and no one is able to make further progress. With many agents placed on a map and a not optimal highway layout, deadlocks can happen quite often, and propagate quickly as the agents in a deadlock could block more agents behind them.

When addressing deadlocks, the first step is to identify whether a deadlock has occurred: The condition that triggers the deadlock identification of each agent is, that it hasn't moved within the last two time steps. The agent then starts a deadlock detection process which recursively builds a chain of agents, each either waiting for the next one to move, or waiting on a permission for a highway modification, starting from itself. The check terminates when it comes across an

agent wanting to move to an unoccupied node and the edge to it lays within the highway graph or, if not, the agent's request to add the edge has been permitted. In this case, no deadlock is detected. On the other hand the algorithm terminates when an already seen agent is reencountered, which indicates a deadlock.

All agents then switch back to the path-planning process, where the agents included within the deadlock replan their paths and decide on new edges to be included in the highway layout. Beforehand, the edge costs from the current vertex to the neighboring ones is modified in each agents' search graphs. If one neighboring vertex is the target of an agent that is not part of the deadlock, the edge will get an infinitely high cost. The same happens with edges of currently occupied vertices regardless of whether the occupying agent is within the deadlock or not. These cost changes will be deleted right after solving the deadlock.

In the event that the A^* -search fails for each agent and no more edges can be added to the highway graph, the latter is deleted, and the algorithm switches once again to the path-planning process, but, this time, with the predefined parameter for w_p and w_r and with all agents contributing. If this, again, does not solve the deadlock, the algorithm is terminated.

Algorithm 2 Path Execution Process

```
1: procedure PATHEXECUTION()  
2:   deadlock ← False  
3:   while my_pos ≠ goal_pos do  
4:     deadlock ← RECEIVE_DEADLOCK_MESSAGES()  
5:     next_pos ← my_path[-1]  
6:     if deadlock then  
7:       SOLVE_DEADLOCK()  
8:     end if  
9:     change_request ← CHECK_NEXT_MOVE(next_pos, hwy_graph)  
10:    SEND_TO_OTHERS(my_pos, next_pos, change_request)  
11:    all_requests ← RECEIVE_REQUESTS()  
12:    my_decisions ← DECIDE_ON_REQUESTS(all_requests)  
13:    SEND_DECISIONS(my_decisions)  
14:    all_decisions ← RECEIVE_DECISIONS()  
15:    UPDATE_HIGHWAY_GRAPH(all_decisions)  
16:    if IS_MOVING_POSSIBLE() then  
17:      my_pos ← my_path.POP()  
18:    else if IS_DEADLOCK() then  
19:      SEND_DEADLOCK_MESSAGE()  
20:    end if  
21:  end while  
22: end procedure
```

4 Experiments

The following chapters describe the experiments performed to analyze the previously introduced algorithm A*+CCHWY. First, the implemented simulation model is described, followed by the used benchmark suite and the experimental settings. Finally, the most important results are identified and analyzed.

4.1 Implementation

The A*+CCHWY, and the algorithms used for the experiments were implemented in Python within a simulation model developed for this purpose.

The implementation of this simulation model was inspired by the agent-based modeling framework MESA [20][26]. Figure 4.1 shows a simple UML diagram of the following described main components.

The model class represents the core of the simulation model and defines what happens during the simulation run. It serves as a container for the other components and holds user-defined attributes, such as the MAPF solver, to be used, including its parameter, the number of agents, the map of the agents' world, the agents' start and goal positions, or similar specifications. Furthermore, it collects the data needed for the bench-marking, such as the sum of cost or the makespan. Each instantiation of the model class serves as a simulation run for a specific benchmark problem and a selected solver.

The agent class describes the agent and its behaviors. Two sub-classes are inherited by the agent class. One class describes dependent agents, which require a central path planner to plan their paths. Their only behavior is to move according to their given paths step by step until reaching their goal. Meanwhile, the second class describes independent, agents which plan their path on their own. The path planner here is included within the class. Furthermore, this class includes methods for simulating the communication capabilities between the agent. For this

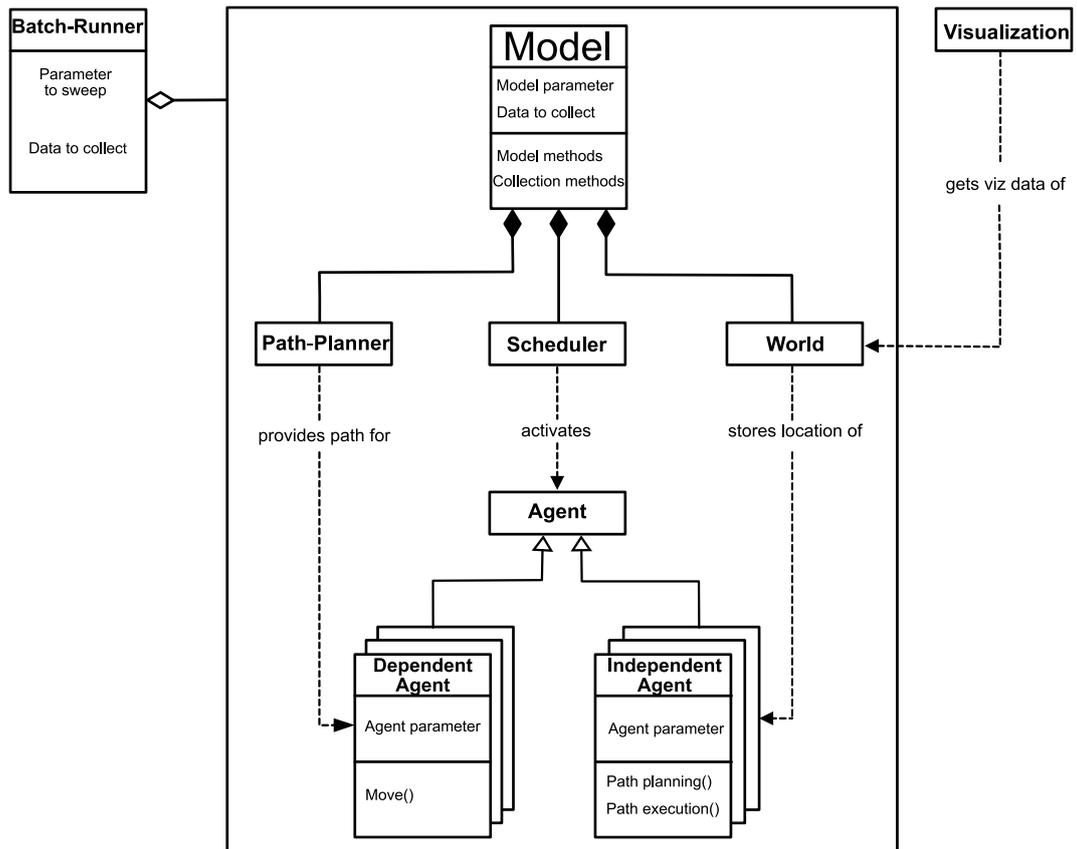


Figure 4.1: Simplified UML model of the simulation model

purpose, public instance variables are defined for each communication topic. All other class attributes aren't access-able from outside the class.

The scheduler module controls the agents' activation regime. At each time step of the model class, the scheduler is called to activate the agents. For this, a staged activation is implemented, allowing the agent activation to be divided into several stages instead of a single step. All agents execute one stage before proceeding to the next.

The world module is an abstraction of the given grid map. It contains information about the map size, the position of obstacles, and, in the case of A*+CCHWY, the highway design. Furthermore, it stores the agents' positions and reports any collisions or illegal movements that occur. In addition, it serves as the basis for the visualization.

The path-planner module is needed only for the agents dependent on a central path planner. It runs the selected MAPF algorithm and provides the planned paths to the appropriate agents.

The batch runner works by generating independent runs for all possible combinations of parameter settings that the user passed to the runner (i.e., the benchmark map, number of agents, type of MAPF solver, solver's parameter setting, and so on). The iterations argument in the batch runner allows for defining how many benchmark scenarios are needed to run a particular combination of settings. Each run terminates after a set time limit or until the model terminates. At that time, the batch runner collects the data needed for the benchmarking from the model class and stores it in a CSV file.

4.2 Problem-Based Benchmark Suite

The benchmark suite used in this work [38] provides grid-based maps of seven categories: real cities, videos games, open grids with or without obstacles, maze-like, room-like, and warehouse-like grids. Each of these maps is assigned to 25 random and even scenarios with a list of paired start and target positions (problems). The random scenarios consist of 1,000 purely random-generated problems. Meanwhile, the even scenarios consist of evenly distributed buckets of random-generated problems with the same length. The following evaluation is made on a selection of these maps, whose characteristics can be traced in Table 4.2, and problems out of the even scenarios.

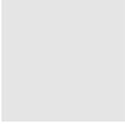
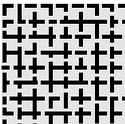
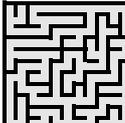
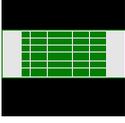
Map	Dimension	#States	Description
Empty 	32x32	1024	No static obstacles
Random 	32x32	922 819	Random 10: 10% obstacles Random 20: 20% obstacles
Room 	32x32	682	64 3×3 rooms connected by 1 single-cell door
Maze 	32x32	666	2-cell-wide corridors
City 	256x256	47540	Map of Berlin Large map with 72% free space
Game 	65 x 81	2445	Map from the game Dragon Age Origin with both wide areas and bottlenecks
Warehouse 	161x63	5699	200 rectangular obstacles building long single-cell-wide corridors

Table 4.1: Characteristics of maps used in the benchmark

4.3 Experimental Design

The A*+CCHWY is analyzed via two main experiments, the first of which is divided into four sub-experiments and the second of which into two sub-experiments. Each experiment yields a data set out of 31 independent experimental runs for each parameter setting and map. The start and target positions of the agents (problems) are taken from the even scenario of the previously presented benchmark suite.

For each experiment, the metrics for the sum of cost (see equation 2.6), the makespan (see equation 2.6), the number of highway changes, and the number of solvable and unsolvable deadlocks are recorded. In addition, two different success rates are computed. The first success rate S_1 represents the number of solved problems in relation to the total number of problems or agent quantity. Meanwhile, the second success rate S_2 is the number of completely solved scenarios in relation to the total number of experimental runs. The experimental parameter setting of the individual experiments is described in the following and can likewise be found in the overview presented in Table 4.3.

4.3.1 Experiment 1: Parameter analysis

Three parameters can be chosen to affect the performance of the A*+CCHWY. On the one hand the weights w_r and w_p to influence the A*-search. On the other hand the weight w_d for weighting the votes within decision making in the path finding process. To examine the impact of these parameters four experiments were performed:

Rewarding the use of highways

The most important parameter for creating an effective highway layout is weight w_r . This rewards edges that conform with the highway layout and therefore determines the level of encouragement for the A*-search to return paths that include such edges.

It is presumed that a smaller w_r during path planning results in fewer changes within the highway layout during path execution and in turn a higher success rate of the algorithm. Furthermore, it is expected that a smaller w_r leads to a deterioration of the makespan and sum of cost. To prove these hypotheses, the following experiments were performed.

Experiment 1.1 runs on the "Empty," "Random10," "Random20," "Room," and "Maze" maps. The agent quantity is set to 64 for the empty and random maps, 56

for the room map, and 40 for the maze map. The 31 independent experimental runs were performed for each $w_r \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and $w_p \in \{1.0, 1.5\}$.

Experiment 1.3 runs on the "Empty," "Random10," and "Random20" maps. The number of agents is set to 40 for each map. The parameters are set as in Experiment 1.1. Since in this experiment all metrics beside the success rate are important, each parameter setting was run until 31 data sets with a success rate of 100% were achieved to avoid distorting the results.

Penalizing conflicting edges

Almost as important as the weight w_r is the parameter w_p . This weight penalizes edges that contradict the highway layout by increasing their costs. It therefore determines the level of encouragement for the A*-search to avoid returning paths that include such edges.

Opposite to w_r , it is presumed that a larger w_p leads to less changes in the highway layout during path execution. This should in turn yield a larger success rate of the algorithm. In addition, it is expected that a larger w_p increases the makespan and sum of cost.

To prove these hypotheses the following experiments were performed:

Experiment 1.2 is constructed similar to Experiment 1.1 but is done for each $w_p \in \{1.0, 1.25, 1.5, 2.0, 3.0, \text{inf}\}$ and $w_r \in \{1.0, 0.5\}$.

Experiment 1.4 is constructed similar to Experiment 1.3 and runs using the same parameter for w_p and w_r as in Experiment 1.2.

Weighted decision making

The weight w_d describes the influence of a single agent's opinion regarding the outcome of the decision of which edge should be added to the highway graph and weights the votes cast for that edge. It is presumed that a weighted decision-making process results in fewer highway changes during path execution than does an unweighted one. To prove this hypothesis, each of the previously described experiments is run for both a decision-making weight $w_d = 1$ and $w_d = w(n, n')$ (see Equation 3.5).

4.3.2 Experiment 2: The Algorithm Performance

This experiment is used to determine the performance of A*+CCHWY with respect to the makespan, sum of cost, and success rate S_2 . For this purpose, A*+CCHWY with $w_p = 1.25$ and $w_r = 0.25$, is compared with three selected path planners, all of which are based on the A* search:

1. Single-agent: Solving MAPF problems by successively executing single agent plans, considering other agents as obstacles. If no path is found for an agent due to other blocking agents, the corresponding agent replan its path at a later time.
2. Priority-based: Each agent is assigned a random priority. The agent paths are planned one after another from the highest-priority agent to the lowest-priority one, considering dynamic obstacles (higher-priority agents).
3. CBS[30]

All MAPF solvers run on the "Empty," map with a time limit of 60s as well as on the "City," "Game," and "Warehouse" maps with a time limit of 600s. The agent quantity varies from 5 to 100 agents for the "Empty" and for 5 to 50 agents for the other maps with a step size of five agents.

4.4 Discussion of Results

The following chapter presents and discusses the most important results of the beforehand described experiments for chosen parameter settings. The parameter settings not included in the discussion produced outcomes with similar effects and were therefore not considered in more detail. All statements made in the evaluation were tested for statistical relevance using a Kruskal-Wallis H test or Mann-Whitney U test with a significance level $\alpha = 0.05$.

4.4.1 Experiment 1: Parameter Analysis

The focus of the analysis was on the "Random20" map, but the presented results can also be applied to the others, including the "Room"; "Empty"; and "Random10" maps. Latter maps only differ in number of obstacles, and they are thus comparable to each other. Moreover, this application to the different maps is possible since the effects of the parameters differ only in their expression and not in their effects.

The maze map, however, is a special case. It turned out that this map was not solvable with the A*+CCHWY for all scenarios independent of the algorithm's parameter settings. This outcome is due to the presence of one-cell-wide corridors and a lack of options for detours for the agents to arrive at their target destinations. In this map, if two agents meet head-to-head in such a corridor, not even

4 Experiments

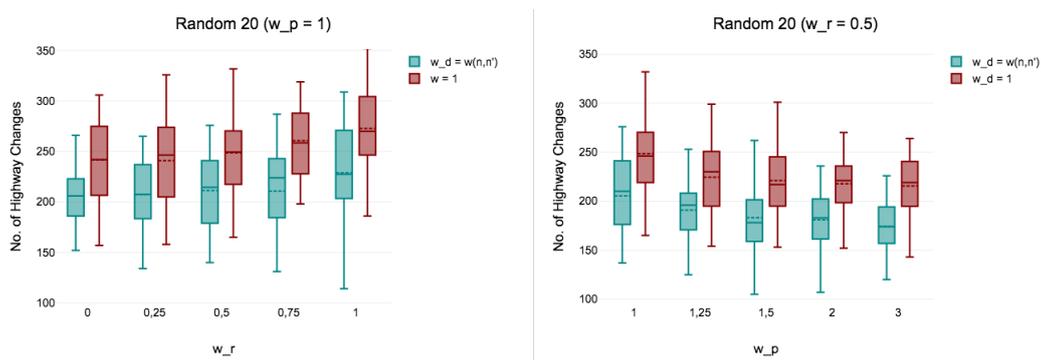


Figure 4.2: Number of highway changes during path execution of A*+CCHWY until MAPF problem of 40 agents was solved on the "Random20" map.

the deadlock solver is of help. Further, if an agent detects that is arriving in a deadlock, it has to re-plan its path. Since detours are not possible within the corridor, the re-planning is not successful for any agent in the deadlock and no edges is added to the highway graph. As a result, the deadlock remains and again the agents in the maze has to re-plan their path. This time, however, the agents can design a whole new highway layout, but even if the agents decide to designate a single direction for movement through a corridor, they will still become stuck in the deadlock since one of the agents would request a change in the highway layout in the next step and the other agent would not agree. After the second deadlock fails to be solved, the algorithm terminates.

Effects on the number of highway changes

Figure 4.2 displays the number of changes in the highway layout during path execution as a result of Experiment 1.3 with $w_p = 1$ and Experiment 1.4 with $w_r = 0.5$.

The boxplots describe the relationship between the level of rewarding or penalizing edges of the search graph within the path planning process and the numbers of changes in the highway layout during path execution.

It is first noticeable that a weighted decision process (blue boxplots) during path planning caused statistically significant fewer changes in the highway layout than the unweighted process (red boxplots) regardless of w_p and w_r . The difference between $w_d = 1$ and $w_d = w(n, n')$ with respect to the median was around 40 changes and corresponded to a moderate correlation of $r = 0.45$.

Furthermore, the number of changes can be seen to have had the tendency to

increase for both a decreasing w_p and an increasing w_r . However, it should be noted that for $w_r = 0.5$ the effect strength approaches zero from $w_p = 1.5$.

Taking these results into account, the following hypotheses could be confirmed:

1. A weighted decision-making process within the path planning process results in fewer changes in the highway layout during path execution.
2. A smaller w_r leads to a smaller number of changes in the highway layout.
3. An increasing w_p decreases the number of changes (with the restriction that the effect strength runs toward zero from $w_r > 1.5$).

It can be concluded that weighted votes lead to better decisions regarding highway layout. The weight w_d influences the decision-making process in the manner that the urgency of each agent whether a certain edge should be added to the highway graph is taken into account. The further away a certain edge is in relation to the agent's position, the more likely the agent has the possibility to re-plan its path. In this case, the weight w_d is smaller than if the edge is directly in front of the agent because then the possibilities to re-plan are less.

In addition, the weights w_r and w_p encourage an implicit cooperative behavior of the agents. The larger w_r is the more the A*-search is influenced by previous decisions and returns paths that contains an increased number of edges from the highway graph. This leads to a further result where the opinion of the agents regarding how a suitable highway layout should look begins to differ less and less from each other. Thus, the agents find a layout solution regarding that is suitable for a large number of agents. This shared solution means fewer changes need to be made during path execution due to vertex conflicts since the agents move in similar ways to a great degree to get to their destination.

The situation is similar with w_p becomes larger, but in this case edge conflicts are avoided instead of vertex conflicts. A larger w_p influences the A*-search to return paths with a decreased number of edges, that would be in conflict with the rules of the highway graph (see Chapter 3.1). Thus, during path execution there are fewer head-to-head conflicts and resulting deadlocks, which solution would lead to changes in the highway layout.

Effects on sum of cost and makespan

Figure 4.3 presents the influence of the rewarding (w_r) and penalizing (w_p) edges of the search graph on the makespan, sum of cost, and the amount of solvable deadlocks during the path planning process. The results are from experiment 1.3, where $w_p = 1.0$, and 1.4 where $w_r = 0.5$. It should be noted that the

4 Experiments

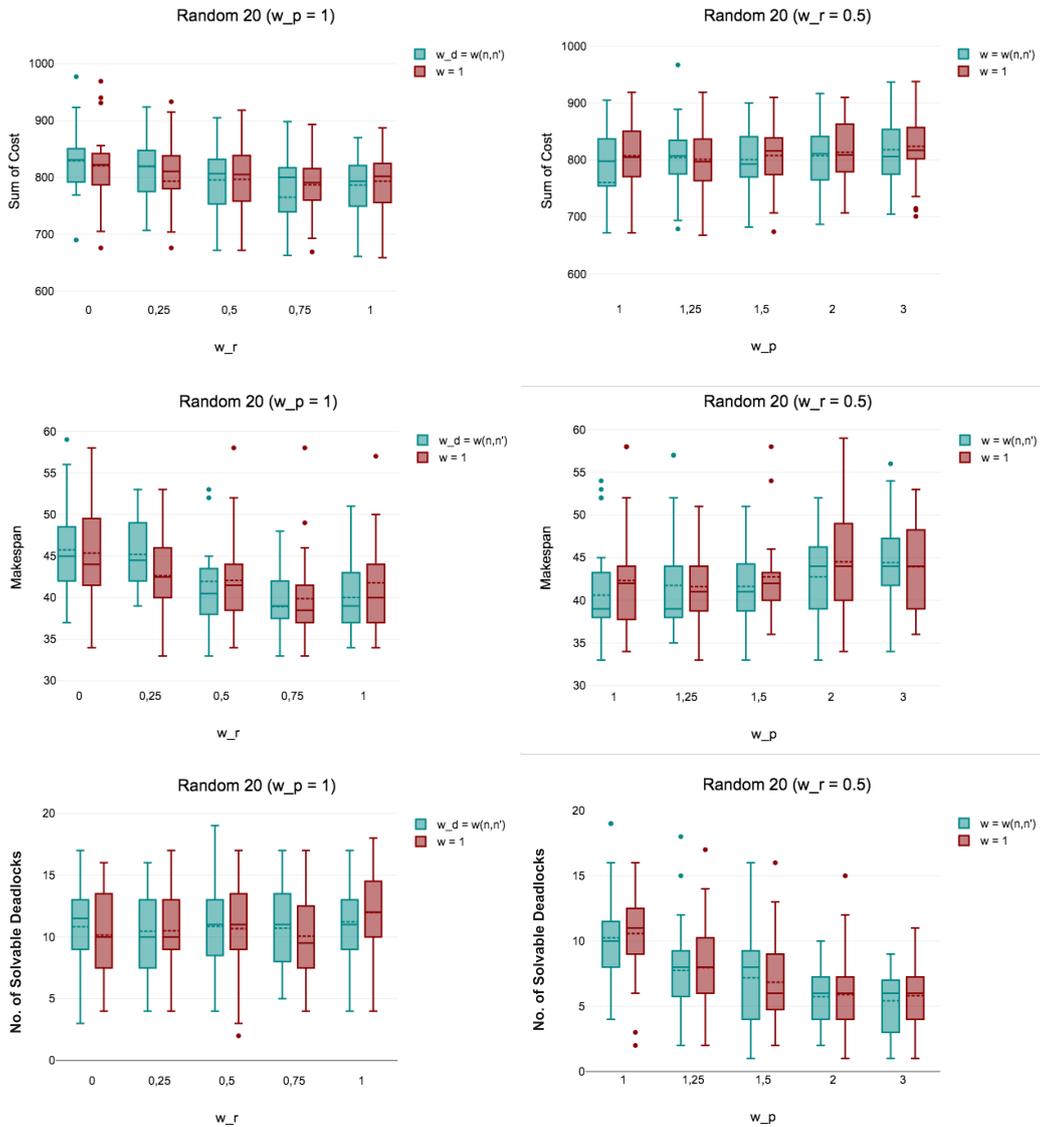


Figure 4.3: MAPF problem of 40 agents was solved on the "Random20" map.

choice of w_d has no statistically significant effect on any of the metrics independent of the choice of w_r and w_p . Due to this lack of effect, w_d is only referred to as a weighted decision-making process with the expression $w_d = w(n, n')$ (see Equation 3.5) in the following.

Concerning the diagrams in Figure 4.3 with a varying w_r , it is noticeable that w_r had no statistically relevant influence on the amount of deadlocks, whereas the median of the sum of cost as well as the makespan decreased with an increasing w_r . Moreover, the strength of the effect was lower for the sum of costs than for the makespan. It should nevertheless be noted that for makespan, the largest effect strength was in the range of $0.25 \leq w_r \leq 0.75$ with a moderate correlation $r = 0.45$. In addition, there was no statically relevant difference with respect to the median for $w_r < 0.25$ as well as for $w_r > 0.75$. For the sum of cost, however, the median decreased almost linearly when increasing w_r with a weak correlation $r = 0.32$.

The results for a varying w_p differed from those for a varying w_r . While the median of the makespan, in the range $1.25 \leq w_p \leq 2$, increases with a greater w_p , the median of the sum of costs remained largely constant for all w_p . Moreover, in contrast to w_r , the number of deadlocks decreased with a greater w_p . It should also be noted here that the effect strength approached zero from $w_p \geq 2$.

In view of the results, the hypothesis that the sum of cost increases with increasing w_p was rejected. In contrast, the following hypotheses were approved with some exceptions:

1. The sum of cost and the makespan deteriorates with a decreasing w_r (only applies for $0.25 \leq w_r \leq 0.75$).
2. The makespan increases with an increasing w_p (only applies for $1.25 \leq w_p \leq 2$).

The effect of w_p and w_r can be explained as follows: The effect of w_p and w_r can be explained as follows. As described before, both parameters influence the A*-search. The higher the w_r or w_p , the higher the degree of influence will be and the further the returned path will deviate from optimality. Due to this deviation from optimality, makespan and flowtime increase as w_r or w_p decreases. That the flowtime nevertheless seems to remain largely constant for a varying w_p is due to the fact that head-to-head conflicts between the agents are avoided with an increasing w_p . The higher the w_p is, the larger the detours of the agents' initial paths are but also the fewer there are of deadlocks. Therefore, the additional detours and waiting times resulting from these deadlocks are also smaller.

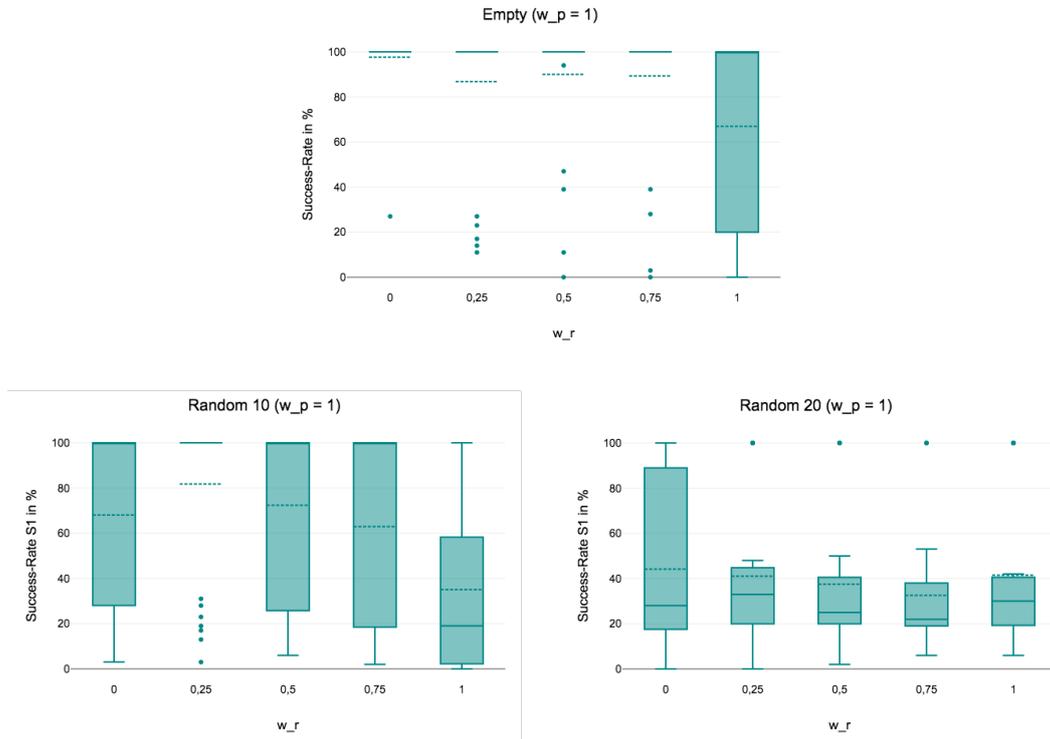


Figure 4.4: Successrate S1 of the A*+CCHWY on the "Empty," "Random 10," and "Random 20," map with $w_p = 1$, $w_d = w(n, n')$ and 64 Agents

Effects on the success-rate

To describe the effects of the parameter settings on the success rate, this section concentrates on the results from experiment 1.1, where $w_p = 1$, and the results from experiment 1.2, where $w_r = 0.5$. In both experiments with $w_d = w(n, n_0)$.

The results of experiment 1.1 are shown in figure 4.4 and 4.5. They show the influence of a varying w_r on the number of problems solved, within an experimental run or scenario, described as success rate S1. When comparing the diagrams for the different maps, it is first noticeable that the success rate deteriorates with an increasing number of obstacles. Upon closer examination of the results for the "Random20" map, one can see that the best success rate, in respect to the median, was reached when $w_r = 0.5$. The success rate decreased with an increasing w_r for $0.25 \leq w_r \leq 0.75$. The cases where $w_r = 0$ and $w_r = 1$ did not fit in the overall picture of a decreasing success rate. When $w_r = 1$, the success rate was too high, and when $w_r = 0$, it was too low.

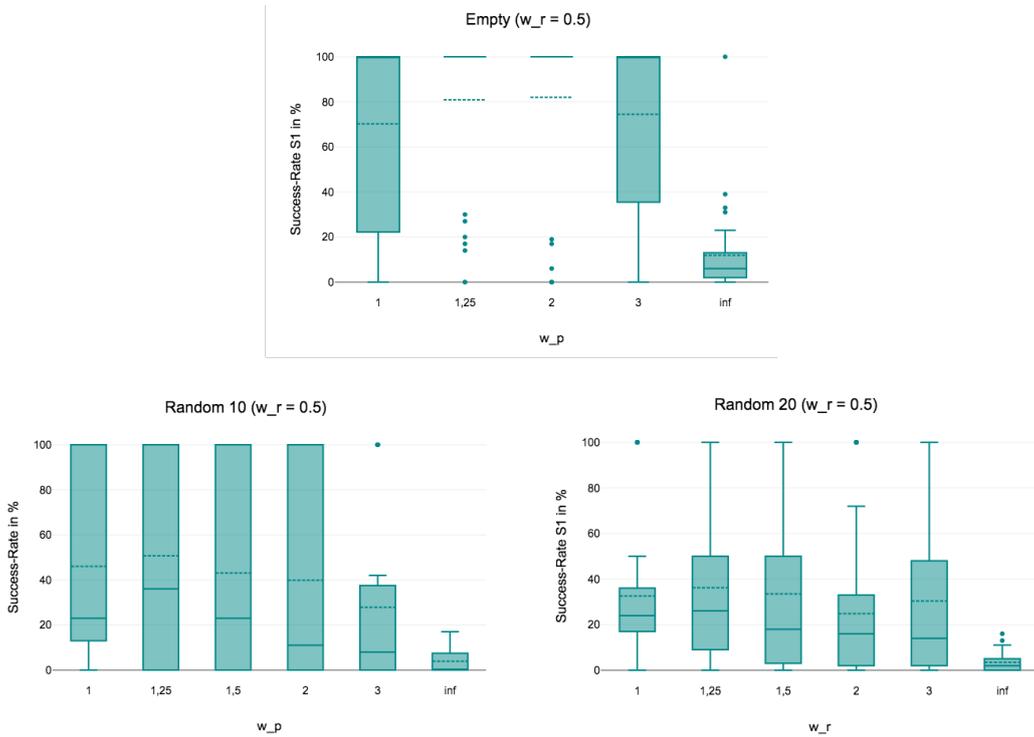


Figure 4.5: Successrate S1 of the A*+CCHWY on the "Empty," "Random 10," and "Random 20" map with $w_r = 0.5$, $w_d = w(n, n')$ and 64 Agents

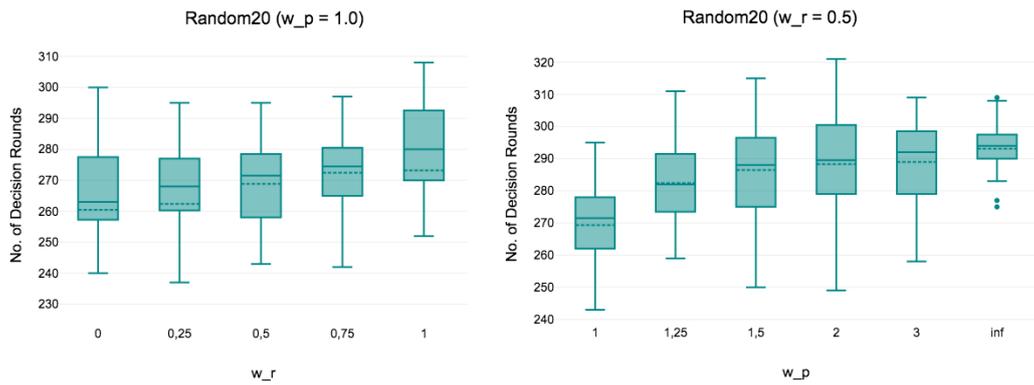


Figure 4.6: No. of decision rounds of the A*+CCHWY during path planning

The results of experiment 1.2 are shown in Figure 4.4, and they reveal the influence of a varying w_p on the success rate $S1$. By comparing the results of each map, it can be seen that in experiment 1.1, the success rate deteriorated when the number of obstacles increased. Focusing on the "Random 10" and "Random 20" maps, it can be further observed that the success rate decreased with an increasing w_p for $1.25 \geq w_p \geq 3$. Notably, the drop for the "Random10" map is significantly stronger than for the "Random20" map. For both maps, however, the effect strength decreases with an increasing w_r .

In response to the results of experiment 1.1 and experiment 1.2, the hypothesis that a smaller w_r as leads to a higher success rate can be approved for $0.75 \leq w_r \leq 0.25$. In addition the hypothesis that a bigger w_r leads to higher success rate can be approved.

The outcomes of the experiments 1.1 and 1.2 can be explained as follows: If the scenario is in general solvable for the A*+CCHWY, the success rate depends on the runtime of the path planning and path execution process during an experimental run with a set time limit.

The runtime of the path planning process in turn depends on the number of decision rounds and the possible path re-planning of the agents. In addition, the duration of the A*-search plays also a role. In the path execution process, the number deadlocks have the most influence on the runtime of the algorithm, since solving deadlocks means to re-plan paths.

Based on this, it can be concluded that the difference in success rates between the three maps is related to the fact that when there is a higher number of obstacles in a map, there is less free space available. As a result, the paths of the agents cross more often and in turn a higher number of decision rounds are needed during path planning as well as a higher quantity of deadlocks during path execution.

The outcomes of the "Random20" map are related to the amount quantity of decision rounds during the path planning process. With a decreasing w_r , the number of decision rounds decreases (see Figure 4.6), This decrease in turn also reduces the runtime of the path planning process. The reason for this is that a decreasing w_r leads to more common opinion among the agents regarding the highway layout due to the influenced A*- search. Since the agents only have to decide on the edges that are included in their opinion, the decision round decreases if the opinions of the agents are similar. That this does not apply for $w_r = 0$ and $w_r = 1$ could be due to the number of expanded nodes during the

A*-search. However, to provide an explanation to this event requires further experiments that were beyond the scope of this thesis. The effect of w_p on success rate can also be explained with the number of decision rounds needed during path planning. As one can see in Figure 4.6 the number of decision rounds increases almost exponentially with an increasing w_p . The result is a higher run time during the path planning process and less remaining time for path execution, which in turn affects the success rate. That $w_p = 1$ deviates from the norm is explained by the resulting higher quantity of deadlock compared to the others. Since deadlocks are solved through path re-planning, they effect the runtime of the path execution and therefore reduce the success rate for $w_p = 1$.

4.4.2 Experiment 2: The Algorithm Performance

The results of experiment 2.1 as well as experiment 2.2 are shown in Figure 4.7. It can be seen that A*+CCHWY worked best on the empty map. There, the success rate only dropped from a number of agents of 60. On the game map, the success rate already fluctuated at 25 agents. However, with a maximum range of 50 agents, the experiment could not show the limit of the algorithm on this map. On the warehouse map, the success rate already dropped at 35 agents and on the city map it performed worst with a decreasing success rate from 10 agents. In comparison to the other algorithms, the A*+CCHWY scaled better as the CBS for all maps. Hereby, it must be said that the CBS couldn't even solve the "Warehouse", "Game" and "City" map with five agents.

In addition the A*+CCHWY outperformed the priority-based planner on the "Empty," "Game" and "City," but not on the "Warehouse" map. Finally even the single-agent planner could be beaten from the A*+CCHWY on the "Game" map, but not on the others.

The advantage of A*+CCHWY over CBS and the priority-based algorithm is that the conflicts regarding the paths of the agents are solved by a time-saving collective decisions. In addition, conflicts are reduced by the influenced A*-search, and resulting common opinion about the highway layout of the agents. This also reduces the runtime of the algorithm and makes it more scalable for a higher number of agents.

The fact that A*+CCHWY performed worse on the warehouse map may be due to the fact that a lot of replanning takes place. Replanning is always necessary when the previously made decision does not conform with the opinion of an agent. Due to the layout of the map, the agents more likely have similar paths. Thus,

4 Experiments

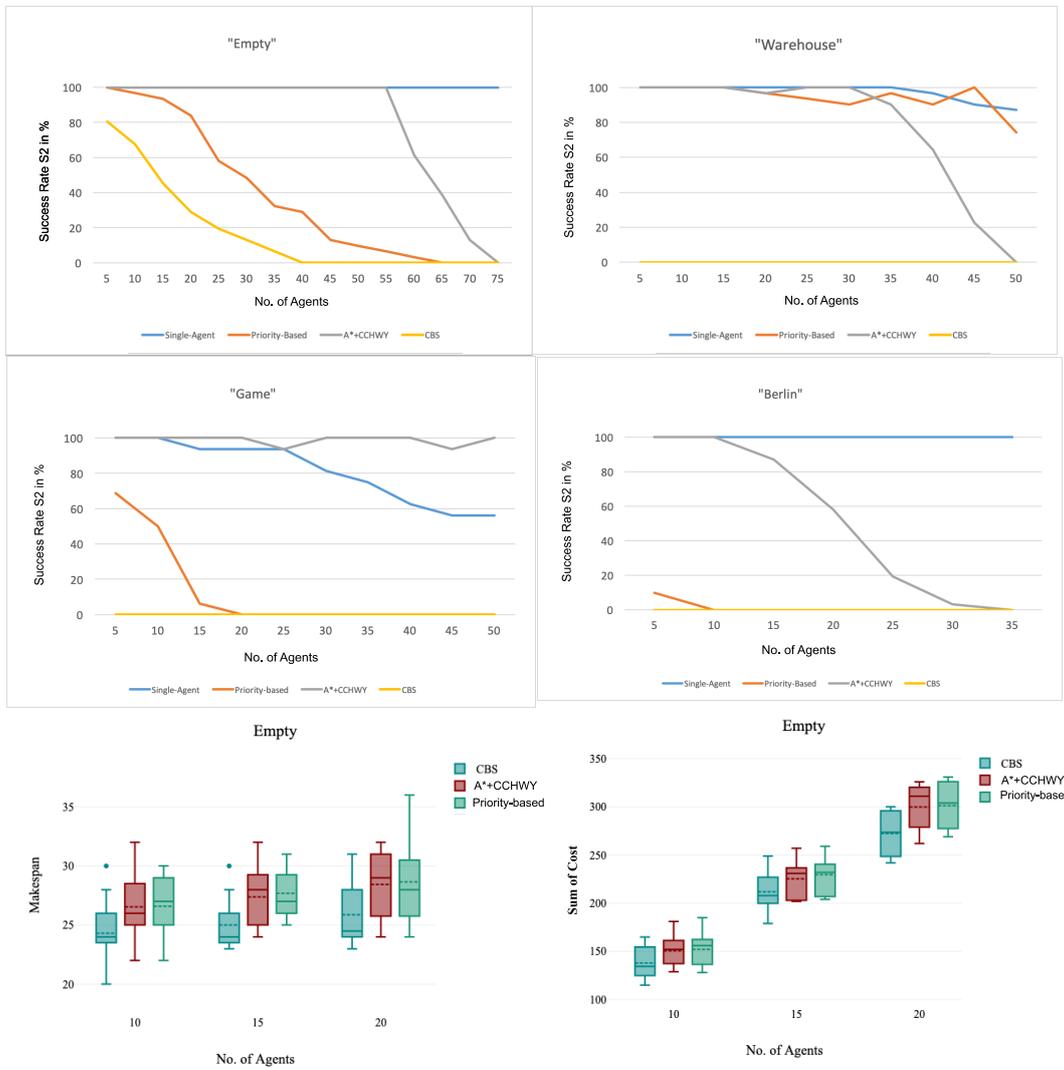


Figure 4.7: Performance of A*+CCHWY in comparison

a decision about an edge that has to be added to the highway graph always involves a high number of agents, all of whom need to replan their path. However, these are only assumptions, which need to be investigated further.

If one looks at the makespan and sum of cost which is displayed exemplary for the "Empty" map in Figure 4.7, it can be seen that the A*+CCHWY performed worse than the CBS and similar or worse than the priority-based planner. Nevertheless, no statement can be made about the suitability of A*+CCHWY, if one only considers the amount of deviation compared to the other planners. It always depends on the situation in which the MAPF solver is used whether a deviation of x time steps is marginal or not.

Even though A*+CCHWY sacrifices optimality, it has been shown that the performance in terms of sum of cost and makespan is still significantly better, than using a single agent planner for a MAPF problem. Following A*+CCHWY is suitable as a MAPF if optimality is not an issue.

At the end it must be said that the A*+CCHWY would perform even better in respect to the success rate if the computing cost will be splitted on the single agents, which is possible due to the design of the algorithm but couldn't be simulated in the experiments.

ID	w_r	w_p	w_d	Map (Agent No.)	Time Limit
1.1	0.0, 0.25, 0.5, 0.75, 1.0	1.0, 1.5	1.0, w(n,n')	Empty (64), Random10 (64), Random20 (64), Room (56), Maze (40)	60s
1.2	1.0, 0.5	1.0, 1.25, 1.5, 2.0, 3.0, inf			
1.3	0.0, 0.25, 0.5, 0.75, 1.0	1.0, 1.5	1.0, w(n,n')	Empty (40), Random10 (40), Random20 (40), Room (40)	inf
1.4	1.0, 0.5	1.0, 1.25, 1.5, 2.0, 3.0, inf			

Table 4.2: Experimental design of Experiment 1: Parameter Analysis

ID	Maps	Agent No.	Time Limit	Solver
2.1	Empty	5-100 (steps of 5)	60s	A*+CCHWY, Single-Agent,
2.2	Warehouse, Game, Berlin	5-50 (steps of 5)	600s	Priority-Based, CBS

Table 4.3: Experimental design of Experiment 2: Algorithm Quality

5 Conclusion and Outlook

In this thesis a new suboptimal approach for multi-agent path finding on grid maps was presented. The algorithm named A*+CCHWY utilizes an A*-based algorithm that prevents collisions by a binding highway layout created through several collective decisions.

Considering the problem topology as a graph with nodes for each grid cell and undirected edges between adjacent cells, then in each decision round, the agents vote for one edge to be added to the highway layout. Additional rules for the voting also ensure that the resulting layout will be designed so that no collisions are possible when moving along the highways. The agents determine their opinion for the decision making process with the help of an A*-search. This A*-search is influenced by the two parameters w_r and w_p . While w_r lowers the cost for edges that are already included in the highway graph, w_p increases the cost for edges that could lead to collisions between the agents when adding to the highway layout. Thus, depending on the settings of these two parameters, the A*-search returns paths that include or do not include such edges in a certain amount of degree. The more the highway graph grows, the more the A*-search is influenced, which in turn leads to a more common opinion among the agents about the design of the highway layout. Therefore, at the end of the decision rounds a highway layout is created that is largely suitable for all agents to reach their destinations. However, to guarantee that each agent arrives at its destination, the agents are allowed to change the highway layout during the path execution process if the other agents agree.

The experiments carried out within the framework of this thesis showed that the choice of parameters to affect the A*-search results in a tradeoff between the quality of the highway layout, the sum of cost or makespan, and the speed of the algorithm. Thus, it became apparent that, except for a few exceptions, a decreasing w_r leads to a better runtime and highway layout but that it also leads to an increased sum of cost and makespan. In contrast, an increasing w_p was found to lead to both a better highway layout and an increased makespan; however, the

run time of the algorithm also expanded. Regarding the sum of cost, it remained the same since the amount of deadlocks and the resulting path re-planning decreased.

The analysis of the parameter revealed in addition, that A*+CCHWY cannot solve problems in maze-like maps with one-cell-wide corridors and is therefore not complete.

Considering the performance compared to other planners, the A*+CCHWY scales significantly better than a CBS. For extremely large maps as well as maps with considerable free space, it also scales better than a simple priority-based planner. Nevertheless, it shows minor weaknesses for warehouse-like domains. Even if A*+CCHWY sacrifices optimality, the results show that its performance in terms of sum of cost and makespan are still significantly better than when a single-agent planner is used for a MAPF problem.

Despite any limitations, the approach presented in this paper demonstrates the potential of solving MAPF through collectively created highways. However, there are many areas that can be examined for potential improvement of the algorithm's performance as well as the suitability of this approach for real-world applications. For example, why the A*+CCHWY scales worse to warehouse-like maps and how to solve this issue could be investigated. In addition, future research could examine the impact a more informed heuristic within the A*-search would have on the algorithm. Furthermore, the impact different opinion definitions or opinion aggregation methods have on the quality of the highway layout and performance of the algorithm could also be examined. Likewise, future research could determine which problems cannot be solved with the A*+CCHWY, such as one cell-wide mazes, and techniques could be developed to tackle these issues. However, the most interesting and important research topics would concern making the algorithm more robust and suitable for real-world applications. This encompasses an investigation into whether and how the algorithm would perform a lifelong MAPF and an examination of how the scope of communication between the agents could be reduced, and how to tackle communication problems, since a complete and faultless communication is currently a prerequisite for the success of the algorithm.

Bibliography

- [1] Kenneth J. Arrow. *Social Choice and Individual Values*. Yale University Press, June 2012.
- [2] David Austen-Smith, John Duggan, and Jeffrey S. Banks, editors. *Social Choice and Strategic Decisions: Essays in Honour of Jeffrey S. Banks*. Studies in Choice and Welfare. Springer, Berlin ; New York, 2005.
- [3] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. *Frontiers in Artificial Intelligence and Applications*, 263:961–962, January 2014.
- [4] Adi Botea and Pavel Surynek. Multi-Agent Path Finding on Strongly Biconnected Digraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015.
- [5] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Eyal Shimony. Icbs: Improved conflict-based search algorithm for multi-agent pathfinding, 2015.
- [6] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, March 2013.
- [7] Scott Camazine, editor. *Self-Organization in Biological Systems*. Princeton Studies in Complexity. Princeton Univ. Press, Princeton, NJ, 2. print., and 1. paperback print edition, 2003.
- [8] E. G. Coffman, M. Elphick, and A. Shoshani. System Deadlocks. *ACM Computing Surveys*, 3(2):67–78, June 1971.
- [9] Liron Cohen and Sven Koenig. Bounded suboptimal multi-agent path finding using highways. pages 3978–3979, January 2016.

- [10] Liron Cohen and Sven Koenig. Bounded suboptimal multi-agent path finding using highways. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16*, page 3978–3979. AAAI Press, 2016.
- [11] Harrie de Swart, Ad van Deemen, Eliora van der Hout, and Peter Kop. Categorical and Ordinal Voting: An Overview. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Harrie de Swart, Ewa Orłowska, Gunther Schmidt, and Marc Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929, pages 147–195. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [12] B. de Wilde, A. W. Ter Mors, and C. Witteveen. Push and Rotate: A Complete Multi-agent Pathfinding Algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, October 2014.
- [13] Esra Erdem, Doga G. Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI'13*, pages 290–296, Bellevue, Washington, July 2013. AAAI Press.
- [14] Ariel Felner, Roni Stern, Eyal Shimony, Eli Boyarski, Meir Goldener, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. page 9, 2017.
- [15] Simon Garnier, Maud Combe, Christian Jost, and Guy Theraulaz. Do Ants Need to Estimate the Geometrical Properties of Trail Bifurcations to Find an Efficient Route? a Swarm Robotics Test Bed. *PLOS Computational Biology*, 9(3):e1002903, March 2013.
- [16] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer. Enhanced Partial Expansion A*. *Journal of Artificial Intelligence Research*, 50:141–187, May 2014.
- [17] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76(12):579–581, December 1989.
- [18] Ko hsin Cindy Wang and Adi Botea. Fast and memory-efficient multi-agent pathfinding. In *In ICAPS*, pages 380–387, 2008.

- [19] M. Jansen and Nathan Sturtevant. Direction maps for cooperative pathfinding. page 6, January 2008.
- [20] Jackie Kazil, David Masad, and Andrew Crooks. Utilizing python for agent-based modeling: The mesa framework. In Robert Thomson, Halil Bisgin, Christopher Dancy, Ayaz Hyder, and Muhammad Hussain, editors, *Social, Cultural, and Behavioral Modeling*, pages 308–317, Cham, 2020. Springer International Publishing.
- [21] Mokhtar M Khorshid, Robert C Holte, and Nathan Sturtevant. A Polynomial-Time Algorithm for Non-Optimal Multi-Agent Pathfinding. page 8, 2011.
- [22] Jens Krause and Graeme D. Ruxton. *Living in Groups*. Oxford Series in Ecology and Evolution. Oxford University Press, Oxford ; New York, 2002.
- [23] Jiaoyang Li, Wheeler Ruml, and Sven Koenig. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. *arXiv:2010.01367 [cs]*, March 2021.
- [24] Ugo Lopez, Jacques Gautrais, Iain D. Couzin, and Guy Theraulaz. From behavioural analyses to models of collective motion in fish schools. *Interface Focus*, 2(6):693–707, December 2012.
- [25] Ryan Luna and Kostas Bekris. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 294–300, January 2011.
- [26] David Masad and Jacqueline Kazil. Mesa: An Agent-Based Modeling Framework. In *Python in Science Conference*, pages 51–58, Austin, Texas, 2015.
- [27] Chris Parker and Hong Zhang. Cooperative Decision-Making in Decentralized Multiple-Robot Systems: The Best-of-N Problem. *Mechatronics, IEEE/ASME Transactions on*, 14:240–251, May 2009.
- [28] Mike Phillips, Benjamin Cohen, Sachin Chitta, and Maxim Likhachev. E-Graphs: Bootstrapping Planning with Experience Graphs. page 8, 2012.
- [29] Chris R. Reid, Hannelore MacDonald, Richard P. Mann, James A. R. Marshall, Tanya Latty, and Simon Garnier. Decision-making without a brain: How an amoeboid organism solves the two-armed bandit. *Journal of The Royal Society Interface*, 13(119):20160030, June 2016.
- [30] Malcolm Ryan. Constraint-based multi-robot path planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 922–928, Anchorage, AK, May 2010. IEEE.

- [31] Alexander Scheidler, Arne Brutschy, Eliseo Ferrante, and Marco Dorigo. The k-Unanimity Rule for Self-Organized Decision Making in Swarms of Robots. *IEEE Transactions on Cybernetics*, 46:1175, May 2016.
- [32] Thomas D. Seeley. *Honeybee Democracy*. Princeton University Press, Princeton, 2010.
- [33] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, February 2015.
- [34] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, February 2013.
- [35] David Silver. Cooperative pathfinding. In *Proceedings of the First AAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AI-IDE’05, page 117–122. AAAI Press, 2005.
- [36] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI’10, page 173–178. AAAI Press, 2010.
- [37] Roni Stern. Multi-Agent Path Finding – An Overview. In Gennady S. Osipov, Aleksandr I. Panov, and Konstantin S. Yakovlev, editors, *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, Lecture Notes in Computer Science, pages 96–115. Springer International Publishing, Cham, 2019.
- [38] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *arXiv:1906.08291 [cs]*, June 2019.
- [39] A. Strandburg-Peshkin, D. R. Farine, I. D. Couzin, and M. C. Crofoot. Shared decision-making drives collective movement in wild baboons. *Science*, 348(6241):1358–1361, June 2015.
- [40] Pavel Surynek. An Optimization Variant of Multi-Robot Path Planning Is Intractable. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1261–1263, July 2010.

- [41] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. *ECAI 2016*, pages 810–818, 2016.
- [42] Gabriele Valentini. How robots in a large group make decisions as a whole? from biological inspiration to the design of distributed algorithms. *arXiv:1910.11262 [nlin, q-bio]*, December 2019.
- [43] Gabriele Valentini, Eliseo Ferrante, and Marco Dorigo. The Best-of-n Problem in Robot Swarms: Formalization, State of the Art, and Novel Perspectives. *Frontiers in Robotics and AI*, 4, 2017.
- [44] Glenn Wagner and Howie Choset. Subdimensional expansion for multi-robot path planning. *Artificial Intelligence*, 219:1–24, February 2015.
- [45] Ko-Hsin Cindy Wang and Adi Botea. MAPP: A Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. page 36.
- [46] Christopher M. Waters and Bonnie L. Bassler. QUORUM SENSING: Cell-to-Cell Communication in Bacteria. *Annual Review of Cell and Developmental Biology*, 21(1):319–346, November 2005.
- [47] Jan Wessnitzer and Chris Melhuish. Collective Decision-Making and Behaviour Transitions in Distributed Ad Hoc Wireless Networks of Mobile Robots: Target-Hunting. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Wolfgang Banzhaf, Jens Ziegler, Thomas Christaller, Peter Dittrich, and Jan T. Kim, editors, *Advances in Artificial Life*, volume 2801, pages 893–902. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [48] Jingjin Yu and Steven M. LaValle. Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation*, pages 3612–3617, May 2013.
- [49] Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI’13*, page 1443–1449. AAAI Press, 2013.

